On the Parameterized Complexity of Grid Contraction

Saket Saurabh
The Institute Of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway
saket@imsc.res.in

Uéverton dos Santos Souza
Fluminense Federal Universidade, Niterói, Brazil
ueverton@ic.uff.br

Prafullkumar Tale
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
prafullkumar.tale@mpi-inf.mpg.de

Abstract

For a family of graphs $\mathcal{G}$, the $\mathcal{G}$-Contraction problem takes as an input a graph G and an integer k, and the goal is to decide if there exists $F \subseteq E(G)$ of size at most k such that G/F belongs to $\mathcal{G}$. Here, G/F is the graph obtained from G by contracting all the edges in F. In this article, we initiate the study of Grid Contraction from the parameterized complexity point of view. We present a fixed parameter tractable algorithm, running in time $c^k \cdot |V(G)|^{\mathcal{O}(1)}$, for this problem. We complement this result by proving that unless ETH fails, there is no algorithm for Grid Contraction with running time $c^{o(k)} \cdot |V(G)|^{\mathcal{O}(1)}$. We also present a polynomial kernel for this problem.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Grid Contraction, FPT, Kernelization, Lower Bound

Digital Object Identifier 10.4230/LIPIcs.SWAT.2020.34

Funding *Saket Saurabh*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 819416), and Swarnajayanti Fellowship (No DST/SJF/MSA01/2017-18).
Prafullkumar Tale: This research is a part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement SYSTEMATICGRAPH (No. 725978).

Acknowledgements Some part of this project was completed when the third author was a Senior Research Fellow at The Institute of Mathematical Sciences, HBNI, Chennai, India. The project started when the second and the third authors were visiting Prof. Michael Fellows at the University of Bergen, Bergen, Norway. Both the authors would like to thank Prof. Michael Fellows for the invitation.

1 Introduction

Graph modification problems are one of the central problems in graph theory that have received a lot of attention in theoretical computer science. Some of the important graph modification operations are vertex deletion, edge deletion, and edge contraction. For graph G, any graph that can be obtained from G by using these three types of modifications is called a *minor* of G. If only the first two types of modification operations are allowed then resulting graph is said to a *subgraph* of G. If the only third type of modification is allowed then the resulting graph is called a *contraction* of G.

For two positive integer r, q, the $(r \times q)$-grid is a graph in which every vertex is assigned a unique pair of the form (i, j) for $1 \leq i \leq r$ and $1 \leq j \leq l$. A pair of vertices (i_1, j_1) and

(i_2, j_2) are adjacent with each other if and only if $|i_1 - i_2| + |j_1 - j_2| = 1$. There has been considerable attention to the problem of obtaining a grid as a minor of the given graph. We find it surprising that the very closely related question of obtaining a grid as a contraction did not receive any attention. In this article, we initiate a study of this problem from the parameterized complexity point of view.

The *contraction* of edge uv in simple graph G deletes vertices u and v from G, and replaces them by a new vertex, which is made adjacent to vertices that were adjacent to either u or v. Note that the resulting graph does not contain self-loops and multiple edges. A graph G is said to be *contractible* to graph H if H can be obtained from G by edge contractions. Equivalently, G is contractible to H if $V(G)$ can be partitioned into $|V(H)|$ many connected sets, called witness sets, and these sets can be mapped to vertices in H such that adjacency between witness sets is consistent with their mapped vertices in H. If such a partition of $V(G)$ exists then we call it H-witness structure of G. A graph G is said to be *k-contractible* to H if H can be obtained from G by k edge contractions. For a family of graphs $\mathcal{G}$, the $\mathcal{G}$-Contraction problem takes as an input a graph G and an integer k, and the objective is to decide if G is k-contractible to a graph H in $\mathcal{G}$.

Related Work. Early papers of Watanabe et al. [20, 21], Asano and Hirata [3] showed $\mathcal{G}$-Contraction is NP-Complete for various class of graphs like planar graphs, outer-planar graphs, series-parallel graphs, forests, chordal graphs. Brouwer and Veldman proved that it is NP-Complete even to determine whether a given graph can be contracted to a path of length four or not [5]. In the realm of parameterized complexity, $\mathcal{G}$-Contraction has been studied with the parameter being the number of edges allowed to be contracted. It is known that $\mathcal{G}$-Contraction admits an FPT algorithm when $\mathcal{G}$ is set of paths [15], trees [15], cactus [17], cliques [6], planar graphs [12] and bipartite graphs [14, 13]. For a fixed integer d, let $\mathcal{H}_{\geq d}, \mathcal{H}_{\leq d}$ and $\mathcal{H}_{=d}$ denote the set of graphs with minimum degree at least d, maximum degree at most d, and d-regular graphs, respectively. Golovach et al. [11] and Belmonte et al. [4] proved that $\mathcal{G}$-Contraction admits an FPT algorithm when $\mathcal{G} \in \{\mathcal{H}_{\geq d}, \mathcal{H}_{\leq d}, \mathcal{H}_{=d}\}$. When $\mathcal{G}$ is split graphs or chordal graphs, the $\mathcal{G}$-Contraction is known to be W[1]-hard [2] and W[2]-hard [18, 6], respectively. To the best of our knowledge, it is known that $\mathcal{G}$-Contraction admits a polynomial kernel only when $\mathcal{G}$ is a set of paths [15] or set of paths or cycle i.e. $\mathcal{H}_{\leq 2}$ [4]. It is known that $\mathcal{G}$ does not admit a polynomial kernel, under standard complexity assumptions, when $\mathcal{G}$ is set of trees [15], cactus [16], or cliques [6].

Our Contribution. In this article we study parameterized complexity of Grid Contraction problem. We define the problem as follows.

Grid Contraction	**Parameter:** k
Input: Graph G and integer k	
Question: Is G k-contractible to a grid?	

To the best of our knowledge, the computation complexity of the problem is not known nor it is implied by the existing results regarding edge contraction problems. We prove that the problem is indeed NP-Complete (Theorem 21). We prove that there exists an FPT algorithm which given an instance (G, k) of Grid Contraction runs in time $4^{6k} \cdot |V(G)|^{\mathcal{O}(1)}$ and correctly concludes whether it is a Yes instance or not (Theorem 20). We complement this result by proving that unless ETH fails there is no algorithm for Grid Contraction with running time $2^{o(k)} \cdot |V(G)|^{\mathcal{O}(1)}$ (Theorem 21). We present a polynomial kernel with $\mathcal{O}(k^4)$ vertices and edges for Grid Contraction (Theorem 27).

Our Methods. Our FPT algorithm for Grid Contraction is divided into two phases. In the first phase, we introduce a restricted version of Grid Contraction problem called Bounded Grid Contraction. In this problem, along with a graph G and an integer k, an input consists of an additional integer r. The objective is to determine whether graph G can be k-contracted to a grid with r rows. We present an FPT algorithm parameterized by $(k+r)$ for this problem. This algorithm is inspired by the exact exponential algorithm for Path Contraction in [1]. It is easy to see that an instance (G, k) is a Yes instance of Grid Contraction if and only if (G, k, r) is a Yes instance of Bounded Grid Contraction for some r in $\{1, 2, \ldots, |V(G)|\}$. In the second phase, given an instance (G, k) of Grid Contraction we produce polynomially many instances of Bounded Grid Contraction such that – (a) the input instance is a Yes instance if and only if at least one of the produced instances is a Yes instance and (b) for any produced instance, say (G', k', r), we have $k' = k$ and $r \in \{1, 2, \ldots, 2k + 5\}$. We prove that all these instances can be produced in time polynomial in the size of the input. An FPT algorithm for Grid Contraction is a direct consequence of these two results. We use techniques presented in the second phase to obtain a polynomial kernel for Grid Contraction.

We present a brief overview of the FPT algorithm for Bounded Grid Contraction. *Boundary vertices* of a subset S of $V(G)$ are the vertices in S which are adjacent to at least one vertex in $V(G) \setminus S$. A subset S of $V(G)$ is *nice* if both $G[S], G - S$ are connected, and $G[S]$ can be contracted to a $(r \times q)$-grid with all boundary vertices in S in an end-column for some integer q. In other words, a subset S of $V(G)$ is nice if it is a union of witness sets appearing in first few columns in some grid witness structure of G. See Definition 9. The objective is to keep building a *special partial solution* for some nice subsets. In this special partial solution, all boundary vertices of a particular nice subset are contained in bags appearing in an end-column. This partial solution is then extended to the remaining graph. The central idea is – *for a nice subset S of graph G, if $G[S]$ can be contracted to a grid such that all boundary vertices of S are in an end bag then how one contract $G[S]$ is irrelevant.* This allows us to store one solution for $G[S]$ and build a dynamic programming table nice subsets of vertices. The running time of such an algorithm depends on the following two quantities (i) the number of possible entries in the dynamic programming table, and (ii) time spent at each entry. We prove that *to bound both these quantities as a function of k, it is sufficient to know the size of neighborhood of S and the size of the union of witness sets in an end-column in a grid contraction of $G[S]$ which contains all boundary vertices of S.*

In the second phase, we first check whether a given graph G can be k-contracted to a grid with r rows for $r \in \{1, 2, \ldots, 2k + 5\}$ using the algorithm mentioned in the previous paragraph. If for any value of r it returns Yes then we can conclude that (G, k) is a Yes instance of Grid Contraction. Otherwise, we argue that there exists a *special* separator S in G which induces a $(2 \times q)$ grid for some positive integer p. We prove that it is safe to contract q vertical edges in $G[S]$. Let G' be the graph obtained from G by contracting these parallel edges. Formally, we argue that G is k-contractible to a $(r' \times q)$-grid if and only if G' is k-contractible to $((r' - 1) \times q)$-grid. We keep repeating the process of finding a special separator and contracting parallel edges in it until one of the following things happens – (a) The resultant graph is k-contractible to a $(r' \times q)$-grid for some $r' < 2k + 5$. (b) The resultant graph does not contain a special separator. We argue that in Case (b), it is safe to conclude that (G, k) is a No instance for Grid Contraction.

Organization of the paper. We present some preliminary notations which will be used in rest of the paper in Section 2. We present a crucial combinatorial lemma in Section 3. As mentioned earlier, this algorithm is divided into two phases. We present the first and the

second phase in Section 4 and 5, respectively. Section 5 also contains an FPT algorithm for Grid Contraction. We prove that the dependency on the parameter in the running time of this algorithm is optimal, up to a constant factor, unless ETH fails in Section 6. In Section 7, we present a polynomial kernel for Grid Contraction problem.

Due to space constraints we have omitted the proofs of the statements marked with ($\star$). We present them in a full version of the paper.

2 Preliminaries

For a positive integer k, $[k]$ denotes the set $\{1, 2, \ldots, k\}$.

2.1 Graph Theory

In this article, we consider simple graphs with a finite number of vertices. For an undirected graph G, sets $V(G)$ and $E(G)$ denote its set of vertices and edges respectively. Two vertices u, v in $V(G)$ are said to be *adjacent* if there is an edge uv in $E(G)$. The neighborhood of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v and its degree $d_G(v)$ is $|N_G(v)|$. The subscript in the notation for neighborhood and degree is omitted if the graph under consideration is clear. For a set of edges F, set $V(F)$ denotes the collection of endpoints of edges in F. For a subset S of $V(G)$, we denote the graph obtained by deleting S from G by $G - S$ and the subgraph of G induced on the set S by $G[S]$. For two subsets S_1, S_2 of $V(G)$, we say S_1, S_2 are adjacent if there exists an edge with one endpoint in S_1 and other in S_2. For a subset S of $V(G)$, let $\Phi(S)$ denotes set of vertices in S which are adjacent with at least one vertex outside S. Formally, $\Phi(S) = \{s \in S |\ N(s) \setminus S \neq \emptyset\}$. These are also called *boundary vertices* of S.

A *path* $P = (v_1, \ldots, v_l)$ is a sequence of distinct vertices where every consecutive pair of vertices is adjacent. For two vertices v_1, v_2 in G, $dist(v_1, v_l)$ denotes the length of a shortest path between these two vertices. A graph is called *connected* if there is a path between every pair of distinct vertices. It is called *disconnected* otherwise. A set S of $V(G)$ is said to be a *connected set* if $G[S]$ is connected. For two vertices v_1, v_2 in G, a set S is called $(v_1\text{-}v_2)$-*separator*, if any v_1-v_2 paths intersects S. If a set is a $(v_1\text{-}v_2)$-separator as well as $(v_3\text{-}v_4)$-separator then we write it as $\{(v_1\text{-}v_2), (v_3\text{-}v_4)\}$-separator.

For two positive integer r, q, the $(r \times q)$-grid is a graph on $r \cdot q$ vertices. The vertex set of this graph consists of all pairs of the form (i, j) for $1 \leq i \leq r$ and $1 \leq j \leq q$. A pair of vertices (i_1, j_1) and (i_2, j_2) are adjacent with each other if and only if $|i_1 - i_2| + |j_1 - j_2| = 1$. We say that such graph is a grid with r rows and q columns. It is called a $(r \times q)$-grid and is denoted by $\boxplus_{r\times q}$. We use $\boxplus$ to denote a grid with unspecified number of rows and columns. The vertices in grid $\boxplus$ are denoted by $\boxplus[i, j]$ or simply by $[i, j]$. Note that the grid with exactly one row is a path. To remove some corner cases, we consider grids that have at least two rows and two columns. Any grid contains exactly four vertices that have degree two. These vertices are called *corner vertices*. Let $t_1 = [1, 1]$, $t_2 = [1, q]$, $t_3 = [r, q]$, and $t_4 = [r, 1]$ be the corner vertices in grid $\boxplus_{r\times q}$.

▶ **Observation 2.1** ($\star$). *If $\hat{S}$ is a connected $\{(t_1\text{-}t_4), (t_2\text{-}t_3)\}$-separator in $\boxplus_{r\times q}$ then its size is at least q. Moreover, if $|\hat{S}| = q$ then it corresponds to a row in $\boxplus_{r\times q}$.*

2.2 Graph Contraction

The *contraction* of edge uv in G deletes vertices u and v from G, and adds a new vertex, which is made adjacent to vertices that were adjacent to either u or v. Notice that no self-loop or parallel edge is introduced in this process. The resulting graph is denoted

by G/e. For a given graph G and edge $e = uv$, we formally define G/e in the following way: $V(G/e) = (V(G) \cup \{w\})\backslash\{u, v\}$ and $E(G/e) = \{xy \mid x, y \in V(G) \setminus \{u, v\}, xy \in E(G)\} \cup \{wx \mid x \in N_G(u) \cup N_G(v)\}$. Here, w is a new vertex which was not in $V(G)$. Note that an edge contraction reduces the number of vertices in a graph by exactly one. Several edges might disappear due to one edge contraction. For a subset of edges F in G, graph G/F denotes the graph obtained from G by contracting each connected component in the sub-graph $G' = (V(F), F)$ to a vertex.

▶ **Definition 1** (Graph Contraction). *A graph G is said to be* contractible *to graph H if there exists an onto function $\psi : V(G) \to V(H)$ such that following properties hold.*

- *For any vertex h in $V(H)$, graph $G[W(h)]$ is connected and not empty, where set $W(h) := \{v \in V(G) \mid \psi(v) = h\}$.*
- *For any two vertices h, h' in $V(H)$, edge hh' is present in H if and only if there exists an edge in G with one endpoint in $W(h)$ and another in $W(h')$.*

We say graph G is contractible to H via mapping ψ. For a vertex h in H, set $W(h)$ is called a *witness set* associated with/corresponding to h. We define *H-witness structure* of G, denoted by $\mathcal{W}$, as collection of all witness set. Formally, $\mathcal{W} = \{W(h) \mid h \in V(H)\}$. A witness structure $\mathcal{W}$ is a partition of vertices in G. If a *witness set* contains more than one vertex then we call it *big* witness-set, otherwise it is *small/singleton* witness set.

If graph G has a H-witness structure then graph H can be obtained from G by a series of edge contractions. For a fixed H-witness structure, let F be the union of spanning trees of all witness sets. By convention, the spanning tree of a singleton set is an empty set. To obtain graph H from G, it is necessary and sufficient to contract edges in F. We say graph G is *k-contractible* to H if cardinality of F is at most k. In other words, H can be obtained from G by at most k edge contractions. The following observations are immediate consequences of definitions.

▶ **Observation 2.2** ($\star$). *If graph G is k-contractible to graph H via mapping ψ then following statements are true.*

1. $|V(G)| \leq |V(H)| + k$.
2. *Any H-witness structure of G has at most k big witness sets.*
3. *For a fixed H-witness structure, the number of vertices in G which are contained in big witness sets is at most $2k$.*
4. *If S is a $(x_1 - x_2)$-separator in G then $\psi(S)$ is a $(\psi(s_1) - \psi(s_2))$-separator in H.*
5. *If S is a separator in G such that there are at least two connected components of $G \setminus S$ which has at least $k + 1$ vertices, then $\psi(S)$ is a separator in H.*

2.3 Parameterized Complexity

An instance of a parameterized problem comprises of an input I, which is an input of the classical instance of the problem and an integer k, which is called as the parameter. A problem Π is said to be *fixed-parameter tractable* or in FPT if given an instance (I, k) of Π, we can decide whether or not (I, k) is a YES instance of Π in time $f(k) \cdot |I|^{\mathcal{O}(1)}$. Here, $f(\cdot)$ is some computable function whose value depends only on k. We say that two instances, (I, k) and (I', k'), of a parameterized problem Π are *equivalent* if $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi$. A *reduction rule*, for a parameterized problem Π is an algorithm that takes an instance (I, k) of Π as input and outputs an instance (I', k') of Π in time polynomial in $|I|$ and k. If (I, k) and (I', k') are equivalent instances then we say the reduction rule is *safe*. A parameterized problem Π admits a kernel of size $g(k)$ (or $g(k)$-kernel) if there is a polynomial

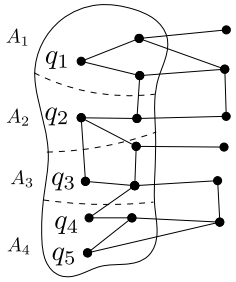

Figure 1 An example of a 4-slab. See Definition 2. For $Q = \{q_1, q_2, q_3, q_4, q_5\}$ and its partition $\mathcal{P}_4(Q) = \{\{q_1\}, \{q_2\}, \{q_3\}, \{q_4, q_5\}\}$, A is an $(\mathcal{P}_4(Q), \alpha, \beta)$-4-slab.

time algorithm (called *kernelization algorithm*) which takes as an input (I, k), and in time $|I|^{\mathcal{O}(1)}$ returns an equivalent instance (I', k') of Π such that $|I'| + k' \leq g(k)$. Here, $g(\cdot)$ is a computable function whose value depends only on k. For more details on parameterized complexity, we refer the reader to the books of Downey and Fellows [8], Flum and Grohe [9], Niedermeier [19], and the more recent books by Cygan et al. [7] and Fomin et al. [10].

3 Combinatorial Lemma

We introduce the notion of r-slabs which can be thought of as connected components with special properties. A r-slab is a connected set which can be partitioned into r connected subsets such that the adjacency between these parts and their neighbourhood follows certain pattern. For an integer r and a set A, an ordered r-partition is a list of subsets of A whose union is A. We define r-slab as follows.

▶ **Definition 2** (r-Slab). *A r-slab in G is an ordered r-partition of a connected set A, say $A_1, A_2, \ldots, A_r$, which satisfy following conditions.*

- *For every i in $[r]$, set A_i is a non-empty set and $G[A_i]$ is connected.*
- *For $i \neq j$ in $[r]$, sets A_i, A_j are adjacent if and only if $|i - j| = 1$.*
- *For every i in $[r]$, define $B_i = N(A_i) \setminus A$. For $i \neq j$ in $[r]$, sets B_i, B_j are mutually disjoint and if B_i and B_j are adjacent then $|i - j| = 1$.*

We denote a r-slab by $\langle A_1, A_2, \ldots, A_r \rangle$. For a r-slab $\langle A_1, A_2, \ldots, A_r \rangle$, set A denotes union of all A_is. We note that every connected subset of G is an 1-slab.

For positive integers α, β, a connected set A in graph G is called an (α, β)-*connected set* if $|A| \leq \alpha$ and $|N(A)| \leq \beta$. For a non-empty set $Q \subseteq V(G)$ a connected set A in G is a (Q)-*connected set* if $Q \subseteq A$. We generalize these notations for r-slab as follows.

▶ **Definition 3** ((α, β)-r-slab). *For a graph G and integers α, β, a r-slab $\langle A_1, A_2, \ldots, A_r \rangle$ is said to be an (α, β)-*r-slab *if $|A| \leq \alpha$ and $|N(A)| \leq \beta$.*

For a set Q, let $\mathcal{P}_r(Q) = \{Q_1, Q_2, \ldots, Q_r\}$ denotes its ordered r-partition. An ordered r-partition is said to be *valid* if for any two vertices $u \in Q_i$ and $v \in Q_j$, u, v are adjacent implies $|i - j| \leq 1$.

▶ **Definition 4** ($\mathcal{P}_r(Q)$-r-slab). *For a graph G, a subset Q of $V(G)$ and its ordered valid partition $\mathcal{P}_r(Q) = \{Q_1, Q_2, \ldots, Q_r\}$, a r-slab $\langle A_1, A_2, \ldots, A_r \rangle$ in G is said to be a $\mathcal{P}_r(Q)$-*r-slab *if Q_i is a subset of A_i for every i in $[r]$.*

See Figure 1 for an example. We combine properties mentioned in previous two definitions to define specific types of r-slabs.

▶ **Definition 5** (**$(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab**). *For a graph G, a non-empty subset Q of $V(G)$, its ordered valid partition $\mathcal{P}_r(Q) = \{Q_1, Q_2, \ldots, Q_r\}$, and integers α, β, a r-slab $\langle A_1, A_2, \ldots, A_r\rangle$ in G is a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab if it is an (α, β)-r-slab as well as a $\mathcal{P}_r(Q)$-r-slab.*

We mention following two observations which are direct consequences of the definition.

▶ **Observation 3.1.** *Let $\langle A_1, A_2, \ldots, A_r\rangle$ be a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab in graph G. If a vertex v is in $N(A)$ then $\langle A_1, A_2, \ldots, A_r\rangle$ is a $(\mathcal{P}_r(Q), \alpha, \beta - 1)$-$r$-slab in graph $G - \{v\}$.*

For a graph G, consider a vertex v and let $G' = G - \{v\}$. For a non-empty subset Q' of $V(G')$, its ordered partition $\mathcal{P}_r(Q') = \{Q'_1, Q'_2, \ldots, Q'_r\}$, and integers α, β, let $\langle A'_1, A'_2, \ldots, A'_r\rangle$ be a $(\mathcal{P}_r(Q'), \alpha, \beta)$-$r$-slab in G'.

▶ **Observation 3.2.** *If vertex v satisfy following two properties then $\langle A'_1, A'_2, \ldots, A'_r\rangle$ is a $(\mathcal{P}_r(Q'), \alpha, \beta + 1)$-$r$-slab in G.*

- *Vertex v is adjacent with exactly one part, say A'_i, of the r-slab*
- *For any vertex u in $N'_G(A'_j) \setminus A'$, if u and v are adjacent in G then $|i - j| \leq 1$.*

Definition 5 generalizes the notation of (Q, α, β)-*connected set* defined in [1]. In the same paper, authors proved that there is an algorithm that given a graph G on n vertices, a non-empty set $Q \subseteq V(G)$, and integers α, β, enumerates all (Q, α, β)-connected sets in G in time $2^{\alpha - |Q| + \beta} \cdot n^{\mathcal{O}(1)}$. We present similar combinatorial lemma for $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs.

▶ **Lemma 6.** *There is an algorithm that given a graph G on n vertices, a non-empty set $Q \subseteq V(G)$, its ordered partition $\mathcal{P}_r(Q) = \{Q_1, Q_2, \ldots, Q_r\}$, and integers α, β, enumerates all $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G in time $4^{\alpha - |Q| + \beta} \cdot n^{\mathcal{O}(1)}$.*

Proof. Let $N(Q) = \{v_1, v_2, \ldots, v_p\}$. Arbitrarily fix a vertex v_l in $N(Q)$. We partition $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G based on whether v_l is contained in it or not. In later case, such $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab is also a $(\mathcal{P}_r(Q), \alpha, \beta - 1)$-$r$-slab in $G - \{v\}$. We now consider the first case. Let i be the smallest integer in $[r]$ such that v_l is adjacent with Q_i. Note that, by definition, if v_l is present in a $\mathcal{P}_r(Q)$-r-slab then it can be part of either A_{i-1}, A_i or A_{i+1}. We encode this fact by moving v_l to either Q_{i-1}, Q_i or Q_{i+1}. Let $\mathcal{P}_r^{i-1}(Q \cup \{v_l\}), \mathcal{P}_r^{i}(Q \cup \{v_l\})$ and $\mathcal{P}_r^{i+1}(Q \cup \{v_l\})$ be r-partitions of $Q \cup \{v_l\}$ obtained from $\mathcal{P}_r(Q)$ by adding v_l to set Q_{i-1}, Q_i and Q_{i+1}, respectively. Formally, these three sets are defined as follows.

- $\mathcal{P}_r^{i-1}(Q \cup \{v_l\}) := \{Q_1, \ldots, Q_{i-1} \cup \{v_l\}, Q_i, Q_{i+1}, \ldots, Q_r\}$
- $\mathcal{P}_r^{i}(Q \cup \{v_l\}) \quad := \{Q_1, \ldots, Q_{i-1}, Q_i \cup \{v_l\}, Q_{i+1}, \ldots, Q_r\}$
- $\mathcal{P}_r^{i+1}(Q \cup \{v_l\}) := \{Q_1, \ldots, Q_{i-1}, Q_i, Q_{i+1} \cup \{v_l\}, \ldots, Q_r\}$

Algorithm. We present a recursive enumeration algorithm which takes $(G, \mathcal{P}_r(Q), \alpha, \beta)$ as an input and outputs a set, say $\mathcal{A}$, of all $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab in G. The algorithm initializes $\mathcal{A}$ to an empty set. The algorithm returns $\mathcal{A}$ if one of the following statements is true: (i) $\mathcal{P}_r(Q)$ is not a valid partition of Q, (ii) $\alpha - |Q| < 0$ or $\beta < 0$, (iii) there is a vertex v_l in $N(Q)$ which is adjacent with Q_i and Q_j for some i, j in $[r]$ such that $|i - j| \geq 2$. If $\alpha - |Q| + \beta = 0$, the the algorithm checks if $\mathcal{P}_r(Q)$ is a $(\mathcal{P}_r(Q), 0, 0)$-r-slabs in G. If it is the case then the algorithm returns singleton set containing $\mathcal{P}_r(Q)$ otherwise it returns an empty set. If there is a vertex v_l in $N(Q)$ which is adjacent with Q_{i-1}, Q_i and Q_{i+1} for some i in $[r]$ then the algorithm calls itself on instance $(G, \mathcal{P}_r^i(Q \cup \{v_l\}), \alpha, \beta)$ where $\mathcal{P}_r^i(Q \cup \{v\})$ is r-partition as defined above. It returns the set obtained on this recursive call as the output. If there

are no such vertices in $N(Q)$, then for some $l \in \{1, \ldots, |N(Q)|\}$, the algorithm creates four instances viz $(G - \{v_l\}, \mathcal{P}_r(Q), \alpha, \beta - 1)$ and $(G, \mathcal{P}_r^{i_0}(Q \cup \{v_l\}), \alpha, \beta)$ for $i_0 \in \{i-1, i, i+1\}$. The algorithm calls itself recursively on these four instances. Let $\mathcal{A}_l^v, \mathcal{A}_l^{i-1}, \mathcal{A}_l^i$, and $\mathcal{A}_{i+1}$ be the set returned, respectively, by the recursive call of the algorithm. The algorithm adds all elements in $\mathcal{A}_l^{i-1} \cup \mathcal{A}_l^i \cup \mathcal{A}_l^{i+1}$ to $\mathcal{A}$. For every $(\mathcal{P}_r(Q), \alpha, \beta - 1)$-$r$-slabs $\langle A'_1, A'_2, \ldots, A'_r \rangle$ in $\mathcal{A}_l^v$, the algorithm checks whether it is a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G using Observation 3.2. If it is indeed a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G then it adds it to $\mathcal{A}$. The algorithm returns $\mathcal{A}$ at the end of this process.

We now argue the correctness of the algorithm. For every input instance $(G, \mathcal{P}_r(Q), \alpha, \beta)$ we define its measure as $\mu((G, \mathcal{P}_r(Q), \alpha, \beta)) = \alpha - |Q| + \beta$. We proceed by the induction hypothesis that the algorithm is correct on any input whose measure is strictly less than $\alpha - |Q| + \beta$. Consider the base cases $\alpha - |Q| + \beta = 0$. In this case, the only possible $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab is $\mathcal{P}_r(Q)$. The algorithm checks this and returns the correct answer accordingly. We consider the case when $\alpha - |Q| + \beta \geq 1$. Every $(\mathcal{P}_r(Q \cup \{v_l\}), \alpha, \beta)$-$r$-slab is also a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab. The algorithm adds a r-slab in $\mathcal{A}_l^v$ to $\mathcal{A}$ only if it is a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G. Hence the algorithm returns a set of $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G. In remaining part we argue that every $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs is enumerated by the algorithm.

By Definition 2, no vertex in closed neighbourhood of a r-slab can be adjacent to two non-adjacent parts of a r-slab. Hence, if there is a vertex v_l in $N(Q)$ which is adjacent with Q_i and Q_j for some i, j in $[r]$ such that $|i - j| \geq 2$ then the algorithm correctly returns an empty set. Suppose there exists a vertex v in $N(Q)$ which is adjacent with Q_{i-1}, Q_i and Q_{i+1} for some i in $[r]$. By Definition 2, any r-slab containing $\mathcal{P}_r(Q)$ must contains v in it. In this case, the number of $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab is same as the number of $(\mathcal{P}_r^i(Q \cup \{v\}), \alpha, \beta)$-$r$-slab where $\mathcal{P}_r^i(Q \cup \{v\})$ is the r-partition of $Q \cup \{v\}$ obtained from $\mathcal{P}_r(Q)$ by adding v to Q_i. The measure for input instance $(G, \mathcal{P}_r^i(Q \cup \{v\}), \alpha, \beta)$ is strictly smaller than $\alpha - |Q| + \beta$. Hence by induction hypothesis, the algorithm correctly computes all $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab.

Consider the case when there is no vertex which is adjacent with Q_{i-1}, Q_i and Q_{i+1} for any i in $[r]$. Let v_l be a vertex in $N(Q)$ and there is an integer i in $[p]$ such that i is the smallest integer, and v_l is adjacent with Q_i. As mentioned earlier, either v_l is a part of $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab or not. In first case, by Definition 2, v_l can be part of A_{i-1}, A_i or A_{i+1} in any $\mathcal{P}_r(Q)$-r-slab. The measure of input instance $(G, \mathcal{P}_r^{i_0}(Q \cup \{v\}), \alpha, \beta)$ is $\alpha - |Q| + \beta - 1$. Hence by induction hypothesis, the algorithm correctly enumerates all $(\mathcal{P}_r^{i_0}(Q \cup \{v\}), \alpha, \beta)$-$r$-slabs in G_l. Consider a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab $\langle A_1, A_2, \ldots, A_r \rangle$ in G which does not contain v_l. By Observation 3.1, $\langle A_1, A_2, \ldots, A_r \rangle$ is a $(\mathcal{P}_r(Q), \alpha, \beta - 1)$-$r$-slab in $G - \{v\}$. By induction hypothesis, the algorithm correctly computes all $(\mathcal{P}_r(Q), \alpha, \beta - 1)$-$r$-slabs in $G - \{v\}$. Since $\langle A_1, A_2, \ldots, A_r \rangle$ is a $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slab in G, vertex v_l satisfy both the properties mentioned in Observation 3.2. Hence algorithm adds $\langle A_1, A_2, \ldots, A_r \rangle$ to the set $\mathcal{A}_l$. Hence, we can conclude that the algorithm correctly enumerates all $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in G

Using the induction hypothesis that the algorithm correctly outputs the set of all $(\mathcal{P}_r(Q), \alpha, \beta)$-$r$-slabs in time $4^{\alpha - |Q| + \beta} \cdot n^{\mathcal{O}(1)}$, the running time of the algorithm follows. This concludes the proof of the lemma. ◀

We use following corollary of Lemma 6.

▶ **Corollary 7.** *There is an algorithm that given a graph G on n vertices and integers α, β, enumerates all (α, β)-r-slab in G in time $4^{\alpha+\beta} \cdot n^{\mathcal{O}(1)}$.*

4 An FPT algorithm for Bounded Grid Contraction

In this section, we present an FPT algorithm for Bounded Grid Contraction. We formally define the problem as follows.

Bounded Grid Contraction	**Parameter:** k, r
Input: Graph G and integers k, r	
Question: Is G k-contractible to a grid with r rows?	

We start with a definition of nice subsets mentioned in the Introduction section. As mentioned before, vertices of a nice subset corresponds to witness sets in first few columns of a grid-witness structure of the input graph. Hence boundary vertices of a nice set corresponds to witness sets in some column of a grid. Note that we are interested in the grids that have exactly r-rows. Hence, we use the notation of r-slab defined in previous section to formally define nice sets. Consider a r-slab $\langle D_1, D_2, \ldots, D_r \rangle$ which corresponds to a column in some grid that can be obtained from the input graph with at most k edge contraction. By Observation 2.2, an edge contraction reduces the number of vertices by exactly one. As there are $3r$ many vertices in three adjacent rows in a grid, the size of closed neighborhood of D in G is at most $k + 3r$. Thus, we can focus our attention on r-slabs with bounded closed neighborhood. We define k-potential r-slabs as follows.

▶ **Definition 8** (k-Potential r-Slab). *For a given graph G and integers k, r, a r-slab $\langle D_1, D_2, \ldots, D_r \rangle$ is said to be a* k-potential r-slab *of G if it satisfies following two conditions:*

- $|D| + |N(D)| \leq k + 3r$*; and*
- $G - D$ *has at most two connected components.*

Here, $D = D_1 \cup D_2 \cup \cdots \cup D_r$.

▶ **Definition 9** (Nice Subset). *A subset S of $V(G)$ is said to be a* nice subset *of G if there exists a k-potential r-slab, say $\langle D_1, D_2, \ldots, D_r \rangle$, such that D is a subset of S and $G[S \setminus D]$ is one of the connected components of $G - D$. We say that r-slab $\langle D_1, D_2, \ldots, D_r \rangle$ is* responsible *for nice subset S.*

Since $\langle D_1, D_2, \ldots, D_r \rangle$ is a k-potential r-slab, both $G[S]$ and $G - S$ are connected. There may be more than one k-potential r-slabs responsible for a nice subset. We define a pair of nice sets and k-potential r-slabs responsible for it.

▶ **Definition 10** (Valid Tuple). *A tuple $(S, \mathcal{P}_r(D))$ is called a* valid tuple *if S is a nice subset and $\mathcal{P}_r(D) \equiv \langle D_1, D_2, \ldots, D_r \rangle$ is a k-potential r-slab responsible for it.*

Let $\mathcal{V}_k$ be the set of all valid tuples. For a valid tuple $(S, \mathcal{P}_r(D))$ in $\mathcal{V}_k$, we define a collection of k-potential r-slabs which is denoted by $\mathcal{A}[(S, \mathcal{P}_r(D))]$. This set can be thought of as a collection of "potential column extenders" for S. See Figure 2. In other words, we can append a k-potential-r-slab in $\mathcal{A}[(S, \mathcal{P}_r(D))]$ to get a grid witness structure of a larger graphs containing S. Let $\mathcal{P}_r(A)$ be a k-potential-r-slab in $\mathcal{A}[(S, \mathcal{P}_r(D))]$. Intuitively speaking, $\mathcal{P}_r(A)$ is the "new" column to be "appended" to a grid witness structure of $G[S]$, to obtain a grid witness structure for $G[S \cup A]$. Hence if $G[S]$ can be k'-contracted to a grid then $G[S \cup A]$ can be $k' + (|A| - r)$-contracted to a grid. For improved analysis, we concentrate on subset $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]$ of $\mathcal{A}[(S, \mathcal{P}_r(D))]$ defined for integers a, b. The set $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]$ is a collection of k-potential r-slabs of size at most a which have at most b neighbors outside S. We impose additional condition that $a + b + |D|$ is at most $k + 3r$ for improved analysis. Formally, $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))] = \{\langle A_1, A_2, \ldots, A_r \rangle \mid |A| \leq a, |N(A) \setminus S| \leq b,$ where $A = A_1 \cup A_2 \cup \cdots \cup A_r$ and for every D_i in $\mathcal{P}_r(D), (N(D_i) \setminus S) \subseteq A_i,$ and $a + b + |D| \leq k + 3r\}$.

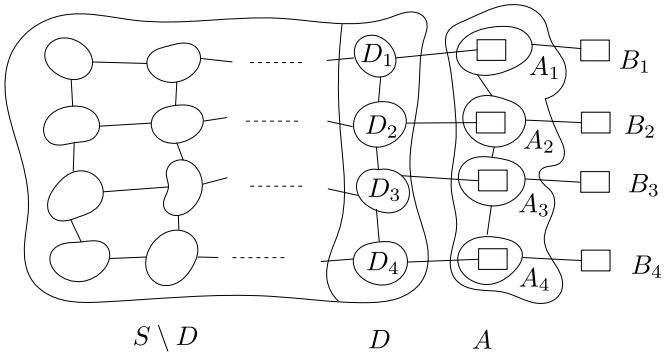

Figure 2 All sets with smooth (non-rectangular) boundary are connected. Set A is a possible extension of nice subset S. In other words, A is an element in $\mathcal{A}_{|A|,|B|}[(S, \mathcal{P}_r(D))]$. See paragraph before Lemma 11.

Algorithm. The algorithm takes a graph G on n vertices and integers k, r as input and outputs either `True` or `False`. The algorithm constructs a dynamic programming table Γ in which there is an entry corresponding to every index $[(S, \mathcal{P}_r(D)); k']$ where $(S, \mathcal{P}_r(D))$ is a valid tuple in $\mathcal{V}_k$ and k' is an integer in $\{0\} \cup [k]$. It initialize values corresponding to all entries to `False`.
(*for-loop Initialization*) For a tuple $(S, \mathcal{P}_r(D)) \in \mathcal{V}_k$ such that $S = D$ and $k' \geq |S| - r = |D| - r$, the algorithm sets $\Gamma[(S, \mathcal{P}_r(D)); k'] =$ `True`.
(*for-loop Table*) The algorithm processes indices in the table in chronologically increasing order. It first checks the size of S, then the size of D, followed by k. Ties are broken arbitrarily. At table index $[(S, \mathcal{P}_r(D)); k']$, if $\Gamma[(S, \mathcal{P}_r(D)); k']$ is `False` then the algorithm continues to next tuple. If $\Gamma[(S, \mathcal{P}_r(D)); k']$ is `True` then it runs the following for-loop at this index.
(*for-loop at Index*) The algorithm computes the set $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]$ for every pair of integers $a\ (\geq r), b\ (\geq 0)$ which satisfy following properties (1) $a + b + |D| \leq k + 3r$, (2) $k' + a - r \leq k$, and (3) $|N(S)| \leq a$. For every k-potential r-slab $\mathcal{P}_r(A)$ in $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]$, the algorithm sets $\Gamma[(S \cup A, \mathcal{P}_r(A)); k_1]$ to `True` for every $k_1 \geq k' + (a - r)$.
If $\Gamma[(V(G), \mathcal{P}_r(D)); k']$ is set to `True` for some $\mathcal{P}_r(D)$ and k' then the algorithm returns `True` otherwise it returns `False`. This completes the description of the algorithm.

Recall that for a given connected subset S of $V(G)$, $\Phi(S)$ denotes its boundary vertices i.e. set of vertices in S which are adjacent with at least one vertex outside S.

▶ **Lemma 11.** *For every tuple $(S, \mathcal{P}_r(D))$ in $\mathcal{V}_k$ and integer k' in $\{0\} \cup [k]$, the algorithm assign $\Gamma[(S, \mathcal{P}_r(D)); k'] =$ `True` if and only if $k' + |N(S)| - r \leq k$ and there is a $(r \times q)$-grid witness structure of $G[S]$, for some integer q, such that $\mathcal{P}_r(D)$ is collection of witness sets in an end-column and $\Phi(S)$ is in D.*

Proof. We prove the lemma by induction on $|S| + k'$ for indices $((S, \mathcal{P}_r(D)); k')$ in the dynamic programming table. For the induction hypothesis, we assume that for a positive integer z the algorithm computes $\Gamma[(S, \mathcal{P}_r(D)); k']$ correctly for each $(S, \mathcal{P}_r(D))$ in $\mathcal{V}_k$ and k' in $0 \cup [k]$ for which $|S| + k' \leq z$.

Consider the base case when $|S| = |D| = r$ and $k' = 0$. Since $D \subseteq S$, we have $S = D$. This implies $\mathcal{P}_r(S) = \mathcal{P}_r(D)$ is a r-slab. Any connected subset of a graph can be contracted to a vertex by contracting a spanning tree. Hence, $G[S]$ can be contracted to a $(r \times 1)$-grid

by contracting $|D| - r$ many edges. This implies that the values assigned by the algorithm in (*for-loop Initialization*) are correct. We note that once the algorithm sets a particular value to `True`, it does not change it afterwards.

Assuming induction hypothesis, we now argue that the computation of $\Gamma[\cdot]$ for indices of the form $[(S_1, \mathcal{P}_r(D_1)); k_1]$ where $|S_1| + k_1 = z + 1$ are correct. Note that if $[(S_1, \mathcal{P}_r(D_1)); k_1]$ is an entry in the table then $(S_1, \mathcal{P}_r(D_1))$ is a valid tuple in $\mathcal{V}_k$ and k_1 is an integer in the set $\{0\} \cup [k]$.

($\Rightarrow$) Assume that $G[S_1]$ is k_1-contractible to a $(r \times q)$-grid such that all vertices in $\Phi(S_1)$ are in an end-column $\mathcal{P}_r(D_1)$ and $k_1 + |N(S_1)| - r \leq k$. We argue that the algorithm sets $\Gamma[(S_1, \mathcal{P}_r(D_1); k_1]$ to `True`. Let $G[S_1]$ be k_1-contractible to a $(r \times q)$-grid. If $q = 1$ then $D_1 = S_1$ and in this case algorithm correctly computes $\Gamma[(S_1, \mathcal{P}_r(D)); k_1]$. Consider the case when $q \geq 2$. Let $\mathcal{W} = \{W_{ij} \mid (i,j) \in [r] \times [q]\}$ be a $(r \times q)$-grid structure of G such that $\mathcal{P}_r(D)$ is collection of witness sets in an end-column and $\Phi(S_1)$ is a subset of D. Define W^c_j as union of all witness sets in column j. Formally, $W^c_j = \bigcup_{i=1}^{r} W_{ij}$. Hence, $\mathcal{W} = W^c_1 \cup W^c_2 \cup \cdots \cup W^c_{q-1} \cup W^c_q$ and $\mathcal{P}_r(D) = W^c_q$. Consider set $S_0 = W^c_1 \cup W^c_2 \cup \cdots \cup W^c_{q-1}$. Since $q \geq 2$, S_0 is an non-empty set. Let $k_0 = k_1 - (|W^c_q| - r)$. We argue that $[(S_0, W^c_{q-1}); k_0]$ is an index in the table and $|S_0| + k_0 \leq z$. As $\mathcal{W}$ is a k_1-grid witness structure, $|W^c_q| - r \leq k_1$ and hence k_0 is a non-negative integer. Since $G[W^c_q]$ is a connected graph, $G - W^c_{q-1}$ has exactly two connected components viz $G[W^c_1 \cup \cdots \cup W^c_{q-2}]$ and the component containing W^c_q. As $\mathcal{W}$ is a k_1-grid witness structure, $|W^c_{q-2}| + |W^c_{q-1}| + |W^c_q| \leq k_1 + 3r \leq k + 3r$ and $N(W^c_{q-1}) \subseteq W^c_{q-2} \cup W^c_q$. (We note that W^c_{q-2} may not exists but this does not change the argument. For the sake of clarity, we do not consider this as separate case.) Since $|W^c_{q-1}| + |N(W^c_{q-1})| \leq k + 3r$ and $G - W^c_{q-1}$ has at most two connected components, W^c_{q-1} is a k-potential r-slab. Note that $\langle W_{1j}, W_{2j}, \ldots, W_{rj} \rangle$ is the r-partition of k-potential r-slab W^c_{q-1}. Hence (S_0, W^c_{q-1}) is a tuple in $\mathcal{V}_k$ and $((S_0, W^c_{q-1}); k_0)$ is an index in the table. Since W^c_q is not an empty set, $|S_0| + k_0 \leq |S_1| - |W^c_q| + k_1 - (|W^c_q| - r) \leq z + 1 + r - 2|W^c_q|$ as $|S_1| + k_1 = z + 1$. Since $|W^c_q| \geq r \geq 1$, we conclude $|S_0| + k_0 \leq z$. Note that $\mathcal{W} \setminus \{W^c_q\}$ is a $(k_1 - |W_q| + r)$-grid witness structure for $G[S_0]$. This implies that $G[S_0]$ is k_0-contractible to a grid with W^c_{q-1} as collection of bags in an end-column and $k_0 + |N(S_0)| - r \leq k_1 \leq k$. Moreover, $S_0 = S_1 \setminus W^c_q$, $\Phi(S_0)$ is contained in W^c_{q-1}. By the induction hypothesis, the algorithm has correctly set $\Gamma[(S_0, W^c_{q-1}); k_0]$ to `True`. Let $x_0 = |W^c_{q-1}|$, $a = |W^c_q|$ and $b = |W^c_q \setminus N(S_0)| = |N(S_1)|$. We first claim that $x_0 + a + b \leq k + 3r$. Note that $|W^c_{q-1}| + |W^c_q| \leq k_1 + 2r$ and $k_1 + b \leq k + r$. Hence $|W^c_{q-1}| + |W^c_q| + b = x_0 + a + b \leq k + 3r$. At index $[(S_0, W^c_{q-1}); k_0]$, the algorithm computes $\mathcal{A}_{a,b}[(S_0, W^c_{q-1})]$. Clearly, W^c_q is one of the sets in $\mathcal{A}_{a,b}[(S_0, W^c_{q-1})]$ as for every i in $[r]$, $N(W_{i,q-1}) \setminus S_0$ is contained in W_{iq} and $G[W_{iq}]$ is a connected graph. Hence the algorithm sets $\Gamma[(S_1, W^c_q), k_1] = \Gamma[(S_1, \mathcal{P}_r(D)), k_1]$ to `True`.

($\Leftarrow$) To prove other direction, we assume that the algorithm sets $\Gamma[(S_1, \mathcal{P}_r(A)); k_1]$ to `True`. We argue that $G[S_1]$ is k_1-contractible to a grid such that $\mathcal{P}_r(A)$ is a collection of witness sets in an end-column in a witness structure; $\Phi(S_1)$ is in A; and $k_1 + |N(S_1)| - r \leq k$. If $\Gamma[(S_1, \mathcal{P}_r(A)); k_1]$ is set to `True` in the (*for-loop Initialization*) then, as discussed in first paragraph, this is correct. Consider the case when the value at $\Gamma[(S_1, \mathcal{P}_r(A)); k_1]$ is set to `True` when the algorithm was processing at index $[(S_0, \mathcal{P}_r(D)); k_0]$. Note that value at $[(S_0, \mathcal{P}_r(D)); k_0]$ has been set `True` by the algorithm as otherwise it will not change any value while processing this index. Note that $|A| = a$ and $|N(S_1)| = b$. Since a is a positive integer and $k_0 + a - r \leq k_1$ (because (*for-loop at Index*) updates only for such values), we know $|S_0| + k_0 \leq |S_1| + k_1 - 2a + r = z + 1 - 2a + r$. Since $a \geq r \geq 1$, we get $|S_0| + k_0 \leq z$. By the induction hypothesis, algorithm has correctly computed value at $[(S_0, \mathcal{P}_r(D)); k_0]$. Hence $G[S_0]$ can be k_0-contracted to a grid such that $\Phi(S_0)$ is in D and there exists a grid witness structure, say $\mathcal{W}_0$, such that $\mathcal{P}_r(D)$ is a collection of witness sets in an end-column. The induction hypothesis also implies and $k_0 + |N(S_0)| - r \leq k + 3r$.

Let $\mathcal{P}_r(A) = \langle A_1, A_2, \ldots, A_r \rangle$ be the r-partition of A in $\mathcal{A}_{a,b}[(S_0, \mathcal{P}_r(A))]$ at which *for-loop at Index* changes the value at $\Gamma[(S_1, \mathcal{P}_r(A)); k_1]$. By construction, every D_i in $\mathcal{P}_r(D)$, D_i is contained in A_i. Since $\Phi(S_0)$ is contained in D, no vertex in $S_0 \setminus D$ is adjacent with any vertex in A. Since $\mathcal{P}_r(A)$ is a r-slab, $\mathcal{W}_0 \cup \{A_1, A_2, \ldots, A_r\}$ is a grid witness structure of $G[S_1]$. Moreover, since $N(S_0)$ is in A, $\Phi(S_1)$ is contained in A. Hence, $G[S_1]$ can be k_1-contractible to a grid with all vertices in $\Phi(S)$ in a A and there exists a witness structure for which $\mathcal{P}_r(A)$ is a collection of witness sets in an end-columns. It remains to argue that $k_1 + |N(S_1)| - r \leq k$. We prove this for the case $k_1 = k_0 + a - r$ as $k_1 > k_0 + a - r$ case follows from the definition of k_1-contratibility. Let $x_0 = |D|$. As x_0 is the size of an end-column in $\mathcal{W}_0$, we have $x_0 - r \leq k_0$. As algorithm only considers a, b such that $x_0 + a + b \leq k + 3r$, substituting $a = k_1 - k_0 + r$ and $b = |N(S_1)|$ we get $x_0 + k_1 - k_0 + r + |N(S_1)| \leq k + 3r$. Using $x_0 - r \leq k_0$, we get the desired bound.

This completes the proof of the lemma. ◀

▶ **Lemma 12.** *Given a graph G on n vertices and integers k, r, the algorithm terminates in time $4^{k+3r} \cdot n^{\mathcal{O}(1)}$.*

Proof. We first describe an algorithm that given a graph G on n vertices and integers k, r, enumerates all valid tuples in time $4^{k+3r} \cdot n^{\mathcal{O}(1)}$. The algorithm computes all r-slabs in G which satisfy first property in Definition 8 using Corollary 7. For every r-slabs, it checks whether it satisfy the second property in Definition 8 to determine whether it is a k-potential r-slab or not. For a k-potential r-slab $\mathcal{P}_r(D) \equiv \langle D_1, D_2, \ldots, D_r \rangle$, if $G - D$ has exactly one connected component, say C_1, the it adds $(V(C_1) \cup D, \mathcal{P}_r(D))$ and $(D, \mathcal{P}_r(D))$ to set of valid tuples. If $G - D$ has two connected components, say C_1, C_2, then it adds $(V(C_1) \cup D, \mathcal{P}_r(D))$ and $(V(C_2) \cup D, \mathcal{P}_r(D))$ to the set of valid tuples. This completes the description of the algorithm. Note that the algorithm returns a set of valid tuples. For a k-potential r-slab $\mathcal{P}_r(D) \equiv \langle D_1, D_2, \ldots, D_r \rangle$, $G - D$ has at most two connected components. Hence any k-potential r-slab is responsible for at most two nice subsets. By definition of nice subsets, for any nice subset there exists a k-potential r-slab responsible for it. Hence the algorithm constructs the set of all valid tuples. The algorithm spends polynomial time for each r-slab it constructs. Hence, the running time of the algorithm follows from Corollary 7.

The algorithm can computes the table and completes *for-loop Initialization* in time $4^{k+3r} \cdot n^{\mathcal{O}(1)}$ using the algorithm mentioned in above paragraph. We now argue that the *for-loop Table* takes $4^{k+3r} \cdot n^{\mathcal{O}(1)}$ time to complete. We partition the set of valid tuples $\mathcal{V}_k$ using the sizes of the neighborhood of connected component and size of r-slab in a tuple. For two fixed integers x, y, define $\mathcal{V}_k^{x,y} := \{(S, \mathcal{P}_r(D)) \in \mathcal{V}_k \mid |D| \leq x \text{ and } |N(S)| \leq y\}$. In other words, $\mathcal{V}_k^{x,y}$ collection of all nice subsets whose neighborhood is of size y and there is a k-potential r-slab of size x responsible for it. Alternatively, $\mathcal{V}_k^{x,y}$ is a collection of k-nice subsets for which there is a (x, y)-r-slab is responsible for it. Since the number of (x, y)-r-slabs are bounded (Corollary 7) and each k-potential r-slab is responsible for at most two nice subsets, $|\mathcal{V}_k^{x,y}|$ is bounded by $4^{x+y} \cdot n^{\mathcal{O}(1)}$.

For each $(S, \mathcal{P}_r(D)) \in \mathcal{V}_k^{x,y}$, the algorithm considers every pair of integers $a(> 0), b(\geq 0)$, such that $x + a + b \leq k + 3$ and $|N(S)| = y \leq a$, and computes the set $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]$. By Lemma 6, set $\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]$ can be computed in time $4^{a+b-|N(S)|} \cdot n^{\mathcal{O}(1)}$. The algorithm spends time proportional to $|\mathcal{A}_{a,b}[(S, \mathcal{P}_r(D))]|$ for *for-loop at Index*. Hence for two fixed integers x, y, algorithm spends

$$\sum_{\substack{a,b \\ x+a+b \leq k+3r}} 4^{x+y} \cdot 4^{a+b-y} \cdot n^{\mathcal{O}(1)} = \sum_{\substack{a,b \\ x+a+b \leq k+3r}} 4^{x+a+b} \cdot n^{\mathcal{O}(1)} = 4^{k+3r} \cdot n^{\mathcal{O}(1)}$$

time to process all valid tuples in $\mathcal{V}_k^{x,y}$. Since there are at most $\mathcal{O}(k^2)$ feasible values for x, y, the overall running time of algorithm is bounded by $4^{k+3r} \cdot n^{\mathcal{O}(1)}$. This concludes the proof. ◀

The following theorem is implied by Lemmas 11, 12, and the fact that $(V(G), \mathcal{P}_r(D))$ is a tuple in $\mathcal{V}_k$ for some D.

▶ **Theorem 13.** *There exists an algorithm which given an instance (G, k, r) of* Bounded Grid Contraction *runs in time $4^{k+3r} \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a* Yes *instance or not. Here, n is the number of vertices in G.*

5 An FPT algorithm for Grid Contraction

In this section, we present an FPT algorithm for Grid Contraction. Given instance (G, k) of Grid Contraction is a Yes instance if and only if (G, k, r) is a Yes instance of Bounded Grid Contraction for some r in $\{1, 2, \ldots, |V(G)|\}$. For $r < 2k + 5$, we can use algorithm presented in Section 4 to check whether given graph can be contracted to grid with r rows or not in FPT time. A choice of this threshold will be clear in the latter part of this section. If algorithm returns Yes then we can conclude that (G, k) is a Yes instance of Grid Contraction. If not then we can correctly conclude that if G is k-contractible to a grid then the resulting grid has at least $2k + 5$ rows. This information allows us to find two *rows* in G which can safely be contracted. We need the following generalized version of Grid Contraction to state these results formally.

Annotated Bounded Grid Contraction	**Parameter:** k, r
Input: Graph G, integers k, r, q, and a tuple (x_1, x_2, x_3, x_4) of four different vertices in $V(G)$	
Question: Is G k-contractible to $\boxplus_{r\times q}$ such that there is a $\boxplus_{r\times q}$-witness structure of G in which the witness sets containing x_1, x_2, x_3, and x_4 correspond to four corners in $\boxplus_{r\times q}$?	

Assume that G is k-contractible to $\boxplus_{r\times q}$ with desired properties via mapping ψ. Let t_1, t_2, t_3, and t_4 be corners in $\boxplus_{r\times q}$ such that $t_1 \equiv [1, 1], t_2 \equiv [1, q], t_3 \equiv [r, q]$, and $t_4 \equiv [r, 1]$. There are 4! ways in which vertices in $\{x_1, x_2, x_3, x_4\}$ can be uniquely mapped to corners $\{t_1, t_2, t_3, t_4\}$. For the sake of simplicity, we assume that we are only interest in the case in which x_1, x_2, x_3, x_4 are mapped to t_1, t_2, t_3, and t_4 respectively. In other words, $\psi(x_i) = t_i$ for all $i \in \{1, 2, 3, 4\}$.

We can modify the algorithm presented in Section 4 obtain an algorithm for Annotated Bounded Grid Contraction problem which is fixed parameter tractable when parameterized by $(k + r)$ $(\star)$.

▶ **Lemma 14.** *There exists an algorithm which given an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$ of* Annotated Bounded Grid Contraction *runs in time $4^{k+3r} \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a* Yes *instance or not. Here, n is the number of vertices in G.*

In the case, when $r < 2k + 5$ the algorithm mentioned in the above lemma is fixed parameter tractable when the parameter is k alone. When $r \geq 2k + 5$, we argue that if $(G, k, r, q, (x_1, x_2, x_3, x_4))$ is a Yes instance then there exists a *horizontal decomposition* of G (Lemma 16). We formally define horizontal decomposition as follows.

▶ **Definition 15** (Horizontally-Decomposible). *Consider an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$ of* Annotated Bounded Grid Contraction. *A graph G is said to be* horizontally-decomposible *if $V(G)$ can be partitioned into four non-empty parts C_{12}, S_u, S_v, and C_{34} which satisfies following properties.*

- *The graphs $G[C_{12}], G[C_{34}]$ are connected and $x_1, x_2 \in C_{12}$, $x_3, x_4 \in C_{34}$.*
- *The graph $G[S_u \cup S_v]$ is a $2 \times q$ grid with S_u, S_v correspond to vertices in its two rows.*
- *C_{12} and C_{34} are the two connected components of $G \setminus (S_u \cup S_v)$.*
- *$N(C_{12}) = S_u$ and $N(C_{34}) = S_v$.*

▶ **Lemma 16** (⋆). *Consider an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$ of* Annotated Bounded Grid Contraction *such that $2k+5 \leq r$. If it is a* Yes *instance then there exists a horizontal decomposition of G.*

Consider an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$, let $(C_{12}, S_u, S_v, C_{34})$ be a horizontal decomposition of G. Reduction Rule 5.1 contracts all the edges across S_u, S_v. Note that in the resulting instance, r is decreased by one.

▶ **Reduction Rule 5.1.** *For an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$, let $(C_{12}, S_u, S_v, C_{34})$ be a horizontal decomposition of G. Let $S_u (= \{u_1, u_2, \ldots, u_q\})$ and $S_v (= \{v_1, v_2, \ldots, v_q\})$. Let G' be the graph obtained from G by contracting all the edges in $\{u_j v_j |\ j \in [q]\}$. Return instance $(G', k, r-1, q, (x_1, x_2, x_3, x_4))$.*

As S_u, S_v are $\{(x_1 - x_4), (x_2 - x_3)\}$-separators in G, by Observation 2.2, sets $\psi(S_u), \psi(S_v)$ are $\{(t_1 - t_4), (t_2 - t_3)\}$-separators in $\boxplus_{r \times q}$. We argue that $\psi(S_u)$ and $\psi(S_v)$ correspond to two consecutive rows and it was safe to contract edges across S_u, S_v.

▶ **Lemma 17** (⋆). *Reduction Rule 5.1 is safe.*

It remains to argue that Reduction Rule 5.1 can be implemented in polynomial time. In Lemma 19, we argue there exists an algorithm that can find a horizontal decomposition, if exists, in polynomial time. We use the following structural lemma to prove the previous statement.

▶ **Lemma 18** (⋆). *Given two adjacent vertices u_1, v_1 in G, there is at most one subset S of $V(G)$ such that (a) $G[S]$ is a $(2 \times q)$ grid, (b) u_1, v_1 are two vertices in the first column of $G[S]$, and (c) each row in S is a separator in G. Moreover, if such a subset exists then it can be found in polynomial time.*

▶ **Lemma 19** (⋆). *There exists an algorithm which given an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$ of* Annotated Bounded Grid Contraction *runs in polynomial time and either returns a horizontal decomposition of G or correctly concludes that no such decomposition exits.*

We are now in a position to present main result of this section.

▶ **Theorem 20.** *There exists an algorithm which given an instance (G, k) of* Grid Contraction *runs in time $4^{6k} \cdot n^{\mathcal{O}(1)}$ and correctly determines whether it is a* Yes *instance or not. Here, n is the number of vertices in G.*

Proof. The algorithm starts with checking whether graph G is k-contractible to a path using the algorithm in [15]. If it is then the algorithm returns Yes else it creates polynomially many instances of Annotated Bounded Grid Contraction by guessing all possible values of r, q, x_1, x_2, x_3, x_4. It processes these instances with increasing values of r. Ties are broken arbitrarily. For $r < 2k + 5$, the algorithm check whether $(G, k, r, q, (x_1, x_2, x_3, x_4))$ is a Yes

instance of Annotated Bounded Grid Contraction using Lemma 14. For $r \geq 2k+5$, the algorithm checks whether there exists a horizontal decomposition of G using Lemma 19. If there exists a horizontal decomposition of G then the algorithm applies Reduction Rule 5.1 to obtain another instance of Annotated Bounded Grid Contraction with a smaller value of r. The algorithm repeats the above step until $r < 2k+5$ or the graph in a reduced instance does not have a horizontal decomposition. In the first case, it checks whether a reduced instance is a Yes instance or not using Lemma 14. In the second case, it continues to the next instance created at the start of the algorithm. The algorithm returns Yes if at least one of the instances of Annotated Bounded Grid Contraction is a Yes instance.

It is easy to see that an instance (G, k) of Grid Contraction is a Yes instance if and only if there exists integers r, q in $\{1, 2, \ldots, |V(G)|\}$ and four vertices x_1, x_2, x_3, x_4 in $V(G)$ such that $(G, k, r, q, (x_1, x_2, x_3, x_4))$ is a Yes instance of Annotated Bounded Grid Contraction. Lemma 17 implies the correctness of the step where the algorithm repeatedly applies Reduction Rule 5.1 and check whether the reduced instance is a Yes instance of Annotated Bounded Grid Contraction or not. Consider an instance $(G, k, r, q, (x_1, x_2, x_3, x_4))$ such that $r > 2k+5$ and there is no horizontal decomposition of G. By Lemma 16, the algorithm correctly concludes that it is a No instance and continues to the next instance. This implies the correctness of the algorithm. The running time of the algorithm is implied by Lemmas 14, 19 and the fact that the algorithm presented in [15] runs in time $2^{k+o(k)} \cdot n^{\mathcal{O}(1)}$. ◀

6 NP-Completeness and Lower Bounds

In this section, we prove that Grid Contraction problem is NP-Complete. We also argue that the dependency on the parameter in the running time of the algorithm presented in Section 5 is optimal, up to constant factors in the exponent, under a widely believed hypothesis. Brouwer and Veldman presented a reduction from Hypergraph 2-Colorability problem to H-Contraction problem [5]. We present a reduction from NAE-SAT problem to Hypergraph 2-Colorability problem. We argue that the reduction used by Brouwer and Veldman can be used to reduce the Hypergraph 2-Colorability problem to Grid Contraction problem. Using these reductions and the fact there is no *sub-exponential* time algorithm for NAE-SAT, we obtain desired results.

▶ **Theorem 21.** *Grid Contraction is* NP*-Complete. Moreover, unless ETH fails, it can not be solved in time* $2^{o(n)}$*, where* n *is the number of vertices in an input graph.*

7 Kernelization

In this section, we present a polynomial kernel for the Grid Contraction problem. In Section 5, we reduced an instance of Grid Contraction to polynomially many instances of Annoted Bounded Grid Contraction such that the original instance is a Yes instance if and only one of these instances is a Yes instance. One can argue that exhaustively application of Reduction Rule 5.1 leads to a *Turing Compression*[1] of the size $\mathcal{O}(k^2)$. We use the similar approach, but with weaker bounds, to obtain a kernel of size $\mathcal{O}(k^4)$.

If the input graph is not connected then we can safely conclude that we are working with a No instance. The following reduction rule checks two more criteria in which it is safe to return a No instance.

[1] Please see, for example, [10, Chapter 22] for formal definition.

▶ **Reduction Rule 7.1.** *For an instance (G, k), if*

- *there exists a vertex in G whose degree is more than $k + 5$, or*
- *there are $6k + 1$ vertices in G whose degrees are more than 5,*

then return a trivial No *instance.*

▶ **Lemma 22** (⋆). *Reduction Rule 7.1 is safe.*

We define $\boxed{k_o = (4k + 8) \cdot (k + 1) + 1}$. Consider an instance (G, k) on which Reduction Rule 7.1 is not applicable. If G has at most $k_o^2 + k + 1$ vertices then we can argue that we have a kernel of the desired size. Consider a case when $|V(G)| \geq k_o^2 + k + 1$. We argue that in this case, if (G, k) is a Yes instance then there exits a large grid separator in a graph G (Lemma 24).

▶ **Definition 23** ($(p \times t)$-grid-separator). *Consider an instance (G, k) of Grid Contraction. A subset S of $V(G)$ is called a $(p \times t)$-grid-separator of G if it has following three properties.*

- $G[S] = \Gamma_{p \times t}$.
- *Graph $G - S$ has exactly two connected components, say C_1 and C_2.*
- $|V(C_1)|, |V(C_2)| \geq k + 1$ *and* $N(C_1) = R_1, N(C_2) = R_p$, *where R_1, R_p are the first and last row in $G[S]$.*

▶ **Lemma 24** (⋆). *Consider an instance (G, k) of Grid Contraction such that $|V(G)| \geq k_o^2 + k + 1$. If (G, k) is a Yes instance then there exists a $((4k + 6) \times t)$-grid-separator in G for some integer t.*

We argue that if there is a large grid that is a separator in G then we can safely contract two consecutive rows in this grid.

▶ **Reduction Rule 7.2.** *For an instance (G, k), let S be a $((4k + 6) \times t)$-grid-separator of G for some integer t. Let $S_u (= \{u_1, u_2, \ldots, u_t\})$ and $S_v (= \{v_1, v_2, \ldots, v_t\})$ be two consecutive internal rows in S. Let G' be the graph obtained from G by contracting all the edges in $\{u_j v_j |\ j \in [q]\}$. Return instance (G', k).*

We prove that the reduction rule is safe along the same line as that of Lemma 17.

▶ **Lemma 25** (⋆). *Reduction Rule 7.2 is safe.*

The following lemma, which is analogous to Lemma 19, is essential to argue that Reduction Rule 7.2 can be applied in polynomial time.

▶ **Lemma 26** (⋆). *There exists an algorithm which given an instance (G, k) of Grid Contraction and integers p, t runs in polynomial time and either returns a $(p \times t)$-grid-separator of G or correctly concludes that no such separator exits.*

We are now in a position to present the main result of the section.

▶ **Theorem 27** (⋆). *Grid Contraction admits a kernel with $\mathcal{O}(k^4)$ vertices and edges.*

References

1. Akanksha Agrawal, Fedor Fomin, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale. Path contraction faster than 2^n. *The 46th International Colloquium on Automata, Languages and Programming (ICALP 2019)*, 2019.
2. Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Split contraction: The untold story. *ACM Transactions on Computation Theory (TOCT)*, 11(3):1–22, 2019.
3. Takao Asano and Tomio Hirata. Edge-Contraction Problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983.
4. Rémy Belmonte, Petr A. Golovach, Pim Hof, and Daniël Paulusma. Parameterized complexity of three edge contraction problems with degree constraints. *Acta Informatica*, 51(7):473–497, 2014.
5. Andries Evert Brouwer and Henk Jan Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11(1):71–79, 1987.
6. Leizhen Cai and Chengwei Guo. Contracting few edges to remove forbidden induced subgraphs. In *International Symposium on Parameterized and Exact Computation*, pages 97–109. Springer, 2013.
7. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
8. Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized complexity*. Springer-Verlag, 2013.
9. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.
10. Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.
11. Petr A Golovach, Marcin Kamiński, Daniël Paulusma, and Dimitrios M Thilikos. Increasing the minimum degree of a graph by contractions. *Theoretical computer science*, 481:74–84, 2013.
12. Petr A. Golovach, Pim van 't Hof, and Daniel Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013.
13. Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Inf. Process. Lett.*, 113(22–24):906–912, 2013.
14. Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27(4):2143–2156, 2013.
15. Pinar Heggernes, Pim Van't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68(1):109–132, 2014.
16. R. Krithika, Pranabendu Misra, Ashutosh Rai, and Prafullkumar Tale. Lossy kernels for graph contraction problems. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016*, pages 23:1–23:14, 2016.
17. R Krithika, Pranabendu Misra, and Prafullkumar Tale. An FPT algorithm for contraction to cactus. In *International Computing and Combinatorics Conference*, pages 341–352. Springer, 2018.
18. Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. In *International Symposium on Parameterized and Exact Computation*, pages 243–254, 2013.
19. Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.
20. Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981.
21. Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the NP-hardness of edge-deletion and-contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983.